Sensor Fusion and Tracking Toolbox™ Release Notes

MATLAB®

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Sensor Fusion and Tracking Toolbox™ Release Notes

Contents

R2019b

R2019a

R2018b

R2019b

Perform track-level fusion using a track fuser

Use `trackFuser` to fuse tracks generated by tracking sensors or trackers and architect decentralized tracking systems.

For more details, see the “Track-to-Track Fusion for Automotive Safety Applications” example.

Track objects using a Gaussian mixture PHD tracker

Use `trackerPHD` with a `gmphd` filter to track point objects and extended objects with designated shapes. With `gmphd`, you can also use rectangular object models (such as `ctrect` and `ctrectmeas`) to track objects of rectangular shape.

For more details, see the “Extended Object Tracking and Performance Metrics Evaluation” example.

Evaluate tracking performance using the OSPA metric

Use `trackOSPAMetric` to evaluate the performance of a tracking system against truth based on the optimal subpattern assignment metric.

For more details, see the “Extended Object Tracking and Performance Metrics Evaluation” example.

Estimate orientation using a complementary filter

You can use `complementaryFilter` to estimate orientation based on accelerometer, gyroscope, and magnetometer sensor data.

For more details, see the “Estimate Orientation with a Complementary Filter and IMU Data” example.

Track objects using tracker Simulink blocks

You can use the GNN tracker and JPDA tracker Simulink® blocks to track objects.

For more details on how to use these two blocks, see these example:

- “Track Vehicles Using Lidar Data in Simulink”
- “Track Closely Spaced Targets Under Ambiguity in Simulink”
- Track Simulated Vehicles Using GNN and JPDA Trackers in Simulink

Features supporting ENU reference frame

By specifying the `'ReferenceFrame'` argument, you can set the output reference frame for the following functions and objects as the ENU (east-north-up) frame. The default reference frame for these functions and objects is the NED (north-east-down) frame.

Features Supporting ENU	Description
`imuSensor`	IMU simulation model
`gpsSensor`	GPS receiver simulation model
`altimeterSensor`	Altimeter simulation model
`ecompass`	Orientation from magnetometer and accelerometer readings
`imufilter`	Orientation from accelerometer and gyroscope readings
`ahrsfilter`	Orientation from accelerometer, gyroscope, and magnetometer readings
`ahrs10filter`	Height and orientation from MARG and altimeter readings
`insfilterMARG`	Estimate pose from MARG and GPS data
`insfilterAsync`	Estimate pose from asynchronous MARG and GPS data
`insfilterErrorState`	Estimate pose from IMU, GPS, and monocular visual odometry (MVO) data
`insfilterNonholonomic`	Estimate pose with nonholonomic constraints
`complementaryFilter`	Orientation estimation from a complementary filter

INS filter name and creation syntax changes

The names of these four INS (inertial navigation system) filters have changed.

Old Name	New Name
MARGGPSFuser	insfilterMARG
AsyncMARGGPSFuser	insfilterAsync
ErrorStateIMUGPSFuser	insfilterErrorState
NHConstrainedIMUGPSFuser	insfilterNonholonomic

Also, the old creation syntaxes, which can create INS filters with new names, will be removed in a future release. The new and recommended creation syntaxes directly create these filters by calling their names.

Old and Discouraged	New and Recommended
filter = insfilter	filter = insfilterMARG
filter = insfilter('asyncimu')	filter = insfilterAsync
filter = insfilter('errorstate')	filter = insfilterErrorState
filter = insfilter('nonholonomic')	filter = insfilterNonholonomic

New examples

This release contains several new examples:

- “Track-to-Track Fusion for Automotive Safety Applications”
- “Simulate a Tracking Scenario Using an Interactive Application”
- “Estimate Orientation with a Complementary Filter and IMU Data”
- “Logged Sensor Data Alignment for Orientation Estimation”
- “Track Vehicles Using Lidar Data in Simulink”
- “Track Closely Spaced Targets Under Ambiguity in Simulink”
- “Track Simulated Vehicles Using GNN and JPDA Trackers in Simulink”
- “Convert Detections to objectDetection Format”
- “Remove Bias from Angular Velocity Measurement”
- “Estimating Orientation Using Inertial Sensor Fusion and MPU-9250”
- “Read and Parse NMEA Data Directly From GPS Receiver”

R2019a

Track objects using a Joint Probabilistic Data Association (JPDA) tracker

Sensor Fusion and Tracking Toolbox includes `trackerJPDA` as an alternative to the existing `trackerGNN` and `trackerTOMHT`. `trackerJPDA` applies a soft assignment where multiple detections can contribute to each track, and balances the robustness and computational cost between `trackerGNN` and `trackerTOMHT`.

For more details on using `trackerJPDA`, see these examples:

- Track Vehicles Using Lidar: From Point Cloud to Track List
- Tracking Closely Spaced Targets Under Ambiguity

Track extended objects using a Probability Hypothesis Density (PHD) tracker

You can use `trackerPHD` to track extended objects using a Gamma Gaussian Inverse Wishart (GGIW) PHD filter, `ggiwphd`. `trackerPHD` creates a multisensor, multiobject tracker utilizing the multitarget PHD filters to estimate the states of the target.

For more details on using `trackerPHD`, see these examples:

- Marine Surveillance Using a PHD Tracker
- Extended Object Tracking

Simulate radar and IR detections from extended objects

To represent a platform’s location as a “spatial extent” instead of a single point, you can use `radarSensor` and `irSensor` to simulate radar and IR detections from extended objects by specifying the `Dimensions` property of `platform`.

For more details on how to simulate radar detections from extended objects, see the `Marine Surveillance` example.

Improve tracker performance for large number of targets

`trackerGNN`, `trackerTOMHT` and `trackerJPDA` enable you to reduce the time required to update the tracker by setting a cost calculation threshold via the

AssignmentThreshold property. This, along with other performance improvements, reduces the processing time when tracking a large number of targets.

For more details, see these examples:

- How to Efficiently Track Large Numbers of Objects
- Tracking a Flock of Birds

Estimate pose using accelerometer, gyroscope, GPS, and monocular visual odometry data

The insfilter can create an error-state Kalman filter suitable for pose (position and orientation) estimation based on accelerometer, gyroscope, GPS, and monocular visual odometry data. To create the error-state Kalman filter, use the 'errorState' input argument.

Estimate pose using an extended continuous-discrete Kalman filter

The insfilter can create a continuous-discrete Kalman filter suitable for pose (position and orientation) estimation based on accelerometer, gyroscope, GPS, and magnetometer input. To create the continuous-discrete Kalman filter, use the 'asyncIMU' input argument.

For more details, see the Pose Estimation From Asynchronous Sensors example.

Estimate height and orientation using MARG and altimeter data

Use ahrs10filter to estimate height and orientation based on altimeter readings and MARG (magnetic, angular rate, gravity) data. Typically, MARG data is derived from magnetometer, gyroscope, and accelerometer readings.

Simulate altimeter sensor readings

Use altimeterSensor to simulate altimeter sensor readings based on a ground-truth position.

Model and simulate bistatic radar tracking systems

`radarSensor`, `radarEmitter`, and `radarChannel` support modeling a radar tracking system with bistatic sensors (physically separated transmitter and receiver), including the effects of signal reflections from the target. To create a bistatic radar sensor, set the `DetectionMode` property of `radarSensor` to `'bistatic'`.

For more details, see the Tracking Using Bistatic Range Detections example.

Correct magnetometer readings for soft- and hard-iron effects

Use `magcal` to determine the coefficients needed to correct uncalibrated magnetometer data. You can correct for soft-iron effects, hard-iron effects, or both.

For more details, see the Magnetometer Calibration example.

Determine Allan variance of gyroscope data

Use `allanvar` to determine the Allan variance of gyroscope data. You can use the Allan variance to set noise parameters on your sensor models.

Generate quaternions from uniformly distributed random rotations

Use `randrot` to generate unit quaternions drawn from a uniform distribution of random rotations.

New application examples

This release contains several new application examples:

- Marine Surveillance Using a PHD Tracker shows how to use a PHD tracker to track extended ship targets with radar detections.
- Track Vehicles Using Lidar: From Point Cloud to Track List shows how to use a JPDA tracker to track vehicles with Lidar detections.
- How to Efficiently Track Large Numbers of Objects.
- Tracking a Flock of Birds.

- Tracking Using Bistatic Range Detections.
- Pose Estimation From Asynchronous Sensors.
- Magnetometer Calibration.
- How to Generate C Code for a Tracker.

R2018b

Version: 1.0

New Features

Single-Hypothesis and Multi-Hypothesis Multi-Object Trackers

Sensor Fusion and Tracking Toolbox provides multi-object trackers that fuse information from various sensors. Use `trackerGNN` to maintain a single hypothesis about the objects it tracks. Use `trackerTOMHT` to maintain multiple hypotheses about the objects it tracks.

Estimation Filters for Tracking

Sensor Fusion and Tracking Toolbox provides estimation filters that are optimized for specific scenarios, such as linear or nonlinear motion models, linear or nonlinear measurement models, or incomplete observability.

Estimation filters include:

Estimate Filters	Description
`trackingABF`	Alpha-beta filter
`trackingKF`	Linear Kalman filter
`trackingEKF`	Extended Kalman filter
`trackingUKF`	Unscented Kalman filter
`trackingCKF`	Cubature Kalman filter
`trackingPF`	Particle filter
`trackingMSCEKF`	Extended Kalman filter in modified spherical coordinates
`trackingGSF`	Gaussian-sum filter
`trackingIMM`	Interacting multiple model filter

Inertial Sensor Fusion to Estimate Pose

Sensor Fusion and Tracking Toolbox provides algorithms to estimate orientation and position from IMU and GPS data. The algorithms are optimized for different sensor configurations, output requirements, and motion constraints.

Inertial sensor fusion algorithms include:

Inertial Sensor Fusion Algorithm	Description
ecompass	Estimate orientation using magnetometer and accelerometer readings.
imufilter	Estimate orientation using accelerometer and gyroscope readings
ahrsfilter	Estimate orientation using accelerometer, gyroscope, and magnetometer readings
insfilter	Estimate position and orientation (pose) using IMU and GPS readings.

Active and Passive Sensor Models

Sensor Fusion and Tracking Toolbox provides active and passive sensor models. You can mimic environmental, channel, and sensor configurations by modifying parameters of the sensor models. For active sensors, you can model the corresponding emitters and channels as separate models.

Sensor models include:

Sensor Model	Description
imuSensor	IMU measurements of accelerometer, gyroscope, and magnetometer
gpsSensor	GPS position, velocity, groundspeed, and course measurements
insSensor	INS/GPS position, velocity, and orientation emulator
monostaticRadarSensor	Radar detection generator
sonarSensor	Active or passive sonar detection generator
irSensor	Infrared (IR) detection generator
radarSensor	Radio frequency detection generator

Trajectory and Scenario Generation

Generate ground-truth trajectories to drive sensor models using the `kinematicTrajectory` and `waypointTrajectory` System objects. Simulate tracking of multiple platforms in a 3-D arena using `trackingScenario`.

Visualization and Analytics

Use `theaterPlot` with `trackingScenario` to plot the ground-truth pose, detections, and estimated pose tracks for multi-object scenarios. Get error metrics for tracks using `trackErrorMetrics`. Analyze and compare the performance of multi-object tracking systems using `trackAssignmentMetrics`.

Orientation, Rotations, and Representation Conversions

The `quaternion` data type enables efficient representation of orientation and rotations. Sensor Fusion and Tracking Toolbox provides the following functions for use with the `quaternion` data type:

Rotations	
`rotateframe`	Quaternion frame rotation
`rotatepoint`	Quaternion point rotation

Representation Conversion	
`rotmat`	Convert quaternion to rotation matrix
`rotvec`	Convert quaternion to rotation vector (radians)
`rotvecd`	Convert quaternion to rotation vector (degrees)
`parts`	Extract quaternion parts
`euler`	Convert quaternion to Euler angles (radians)
`eulerd`	Convert quaternion to Euler angles (degrees)
`compact`	Convert quaternion array to N-by-4 matrix

Metrics and Interpolation	
dist	Angular distance in radians
norm	Quaternion norm
meanrot	Quaternion mean rotation
slerp	Spherical linear interpolation

Initialization and Convenience Functions	
ones	Create quaternion array with real parts set to one and imaginary parts set to zero
zeros	Create quaternion array with all parts set to zero
classUnderlying	Class of parts within quaternion
normalize	Quaternion normalization

Mathematics	
times, .*	Element-wise quaternion multiplication
mtimes, *	Quaternion multiplication
prod	Product of a quaternion array
minus, -	Quaternion subtraction
uminus, -	Quaternion unary minus
conj	Complex conjugate of quaternion
ldivide, .\	Element-wise quaternion left division
rdivide, ./	Element-wise quaternion right division
exp	Exponential of quaternion array
log	Natural logarithm of quaternion array
power, .^	Element-wise quaternion power

Array Manipulation	
ctranspose, '	Complex conjugate transpose of quaternion array
transpose, .'	Transpose of quaternion array

Sensor Fusion and Tracking Examples

The release of Sensor Fusion and Tracking Toolbox includes the following examples.

Applications
Air Traffic Control
Multiplatform Radar Detection Fusion
Passive Ranging Using a Single Maneuvering Sensor
Tracking Using Distributed Synchronous Passive Sensors
Search and Track Scheduling for Multifunction Phased Array Radar
Extended Object Tracking
Visual-Inertial Odometry Using Synthetic Data
IMU and GPS Fusion for Inertial Navigation

Multi-Object Trackers
Multiplatform Radar Detection Fusion
Tracking Closely Spaced Targets Under Ambiguity
Tracking Using Distributed Synchronous Passive Sensors
Extended Object Tracking
Introduction to Using the Global Nearest Neighbor Tracker
Introduction to Track Logic

Estimation Filters
Tracking Maneuvering Targets
Tracking with Range-Only Measurements
Passive Ranging Using a Single Maneuvering Sensor

Inertial Sensor Fusion
Estimate Orientation Through Inertial Sensor Fusion
IMU and GPS Fusion for Inertial Navigation
Estimate Position and Orientation of a Ground Vehicle

Inertial Sensor Fusion
Estimate Orientation and Height Using IMU, Magnetometer, and Altimeter

Sensor Models
Inertial Sensor Noise Analysis Using Allan Variance
Simulating Passive Radar Sensors and Radar Interferences
Introduction to Simulating IMU Measurements
Introduction to Tracking Scenario and Simulating Radar Detections
Scanning Radar Mode Configuration

Trajectory and Scenario Generation
Introduction to Tracking Scenario and Simulating Radar Detections
Benchmark Trajectories for Multi-Object Tracking
Multiplatform Radar Detection Generation

Quaternion Representation
Rotations, Orientation and Quaternions
Lowpass Filter Orientation Using Quaternion SLERP